The three body problem solution using Runge-Kutta method

Juan Urrutia-Mustapha Bousackla, Universitat Autònoma de Bercelona

January 18, 2019

Abstract

In order to solve the 3 body problem, it's necessary to compute the movement via a numerical method such as Runge-Kutta. Using these method, it's shown the movement of the the Earth, Sun and an asteroid.

1 Introduction

The three body problem has been an important challenge in physics history. The first one to come up with a possible solution was Leonhard Euler who found three families of periodic solutions as long as the three masses were on the same line. About twenty years later on 1772, Lagrange found a family of stable orbits for the three masses, made of ellipses. As response to the 1884 competition that Oscar II of Sweden made for solving the problem, Henri Poincaré answered with the recurrence theorem which made him won. The theorem states that for certain systems after a big but finite time they advance to the initial state. It was not until 1912 that Sundman found a convergent solution for systems that have a non zero angular momentum. But the solution is not for practical use. The most used method today is a complex numerical method.

2 Numerical method

2.1 Normalization

In order to avoid the truncation error and use more of the computer's memory the variables are normalized using the third Kepler law applied to the Earth Sun system:

$$T^2 = \frac{4\pi R^3_{earth-sun}}{GM_{sun}} \tag{1}$$

And the Newton's gravity law:

$$\vec{F}_g = G\frac{m_1 m_2}{\|r_1 - r_2\|^3}(\vec{r_2} - \vec{r_1}) \tag{2}$$

By combining both equations it's obtained:

$$\vec{F'} = 4\pi^2 \frac{m'_2}{\|r'_1 - r'_2\|^3}(\vec{r'_2} - \vec{r_1}) \tag{3}$$

Using the following normalization:

$$m' = \frac{m}{M_{solar}}$$
$$t' = \frac{t}{T_{solar}}$$
$$x' = \frac{x}{R_{solar}}$$

2.2 Runge-Kutta

The use of simpler method as Euler will give too much error. The Runge-kutta method is used for solving differential equations as ec(3). It's needed to transforms those ecuations in something like this:

$$\begin{aligned} y' &= f(t, y) \\ y(t_o) &= y_o \end{aligned}$$

In every h time step we define the k_1, k_2, k_3 and k_4 as follows.

$$\begin{aligned} k_1 &= f(t, y) \\ k_2 &= f(t + \frac{h}{2}, y + k_1 \frac{h}{2}) \\ k_3 &= f(t + \frac{h}{2}, y + k_2 \frac{h}{2}) \\ k_4 &= f(t + \frac{h}{2}, y + k_2 h) \end{aligned}$$

So to compute the solution one step forward, we define that:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{4}$$

2.3 Runge-Kutta applied to 3 body problem

The formulation for the three body problem applies in the following way:

$$v^i_{t+1} = v^i_t + \frac{dt}{6}(k^i_1 + 2k^i_2 + 2k^i_3 + k^i_4) \tag{5}$$

$$r^i_{t+1} = r^i_t + \frac{dt}{6}(K^i_1 + 2K^i_2 + 2K^i_3 + K^i_4) \tag{6}$$

$$\tag{7}$$

For every time step it's necessary to compute every K and k in the following way. Where the sub index "i" refers to the component on the 3 dimensional space.

$$\begin{bmatrix} v^i_t = k^i_1 \\ a^i_t(r_i) = K^i_1 \end{bmatrix} \rightarrow \begin{bmatrix} v^i_t + a^i_t(r_i)\frac{dt}{2} = k^i_2 \\ a^i_t(r_i + \frac{v^t_i}{2} dt) = K^i_2 \end{bmatrix} \rightarrow \begin{bmatrix} k^i_2 + K^i_2 \frac{dt}{2} = k^i_3 \\ a^i_t(r_i + \frac{k^i_2}{2} dt) = K^i_3 \end{bmatrix} \rightarrow \begin{bmatrix} k^i_3 + K^i_3 \frac{dt}{2} = k^i_4 \\ a^i_t(r_i + k^i_3 dt) = K^i_4 \end{bmatrix}$$

Consult the complete code on the annex.

3 Earth Sun and asteroid system

3.1 Data

Using the normalized values for the system it's obtained:

$$\begin{aligned} m'_{earth} &= 3 * 10^{-6} \\ r'_{earth} &= 1 \end{aligned}$$

And the following data for the asteroid:

$$\begin{aligned} v'_{max} &= 11.59 \\ r'_{min} &= 0.6 \end{aligned}$$

3.2 Results and precision

The accumulative error of the 4 orden Runga-Kutta method is of $O(dt^4)$. In the system earth sun asteroid it's obtained and error of about 0.0005 in one year(Fig.2). This is the result obtained for an arbitrary asteroid

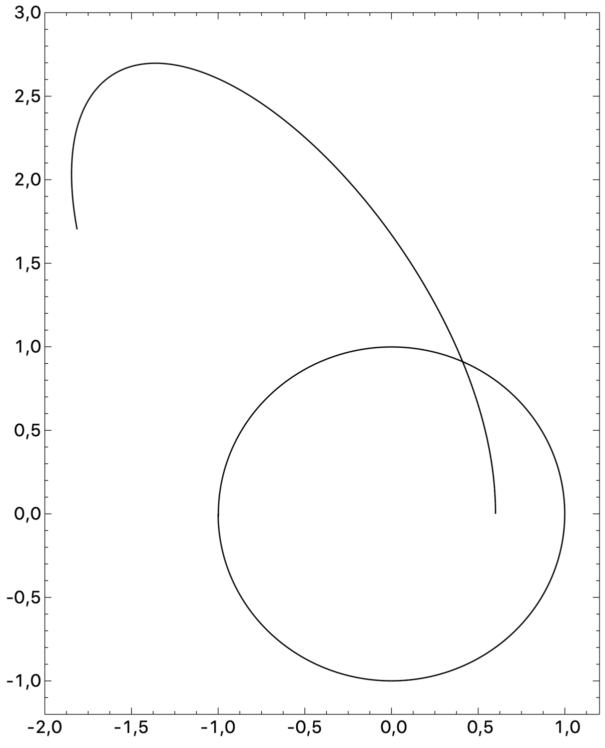

Figure 1: The result for an arbitrary asteroid

with the specified characteristics. We can see the movement of the earth along the sun in one year and the eccentric orbit of the asteroid.

3.3 Conclusions

Runge-Kutta is a good method for solving newton's gravity. By changing the original data of the program developed it's possible to solve many problems such as: the moon-earth-sun system, the Kepler asteroid movement or the Oumuamua approximation to earth. Due to the needed of calculating many parameters in every time iteration it's difficult but not impossible to generalize the 3 body problem to the n-body problem via this method.

4 Code

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <fstream>
#include <iostream>
//Where n is the numbre of iterations
#define n 10000
//These are the mases
#define m1 1
#define m2 0.000003
#define m3 0
//Time is normalized to 1 year
#define t  1.0
int main(){
    std::ofstream fout;
    fout.open ("tres_cuerpos.txt");
    double G;
    int i;
    int j;
    G=4*(3.14159265)*(3.14159265);
    //In here we define the initial conditions
    double dt=t/n;
    double r1[3][n+1];
    double r2[3][n+1];
    double r3[3][n+1];
    double v1[3][n+1];
    double v2[3][n+1];
    double v3[3][n+1];
    double d1d2[n+1];
    double d1d3[n+1];
    double d2d3[n+1];
    double k1[3][3];
    double k2[3][3];
    double k3[3][3];
    double k4[3][3];
    double K1[3][3];
    double K2[3][3];
    double K3[3][3];
    double K4[3][3];
    //The position of the first body
    r1[0][0]=0;
    r1[1][0]=0;
    r1[2][0]=0;
    //The position of the second body
    r2[0][0]=1;
    r2[1][0]=0;
    r2[2][0]=0;
    //The position of the third body
    r3[0][0]=1.00026;
    r3[1][0]=0;
    r3[2][0]=0;
    //Velocity of the first body
```

```
v1[0][0]=0;
v1[1][0]=0;
v1[2][0]=0;
//Velocity of the second body
v2[0][0]=0;
v2[1][0]=2*(3.141516);
v2[2][0]=0;
//Velocity of the third body
v3[0][0]=0;
v3[1][0]=0;
v3[2][0]=0;
//These are the titles for the file
fout << std::fixed;
fout << "    Three body problem" <<std::endl;
fout << "            " <<std::endl;
//Column names
fout << "Position m2(x)                    Position m2(y)" <<std::endl;
//Temporal iteration
for (i=0; i<n; i++) {
    //What
    fout <<r2[0][i]<<"                    "<<r2[1][i]<<std::endl;
    //These is the K for the first body
    //K1
    d1d2[i]=pow(pow(r1[0][i]-r2[0][i], 2)+pow(r1[1][i]-r2[1][i], 2)+pow(r1
        [2][i]-r2[2][i],2),3/2);
    d1d3[i]=pow(pow(r1[0][i]-r3[0][i], 2)+pow(r1[1][i]-r3[1][i], 2)+pow(r1
        [2][i]-r3[2][i], 2),3/2);
    for (j=0; j<=2; j++) {
        K1[j][0]=G*m2/d1d2[i]*(r2[j][i]-r1[j][i])+G*m3/d1d3[i]*(r3[j][i]-
            r1[j][i]);
        k1[j][0]=v1[j][i];
    }
    //K2
    d1d2[i]=pow(pow(r1[0][i]+k1[0][0]*dt/2-r2[0][i], 2)+pow(r1[1][i]+k1
        [1][0]*dt/2-r2[1][i], 2)+pow(r1[2][i]+k1[2][0]*dt/2-r2[2][i],2)
        ,3/2);
    d1d3[i]=pow(pow(r1[0][i]+k1[0][0]*dt/2-r3[0][i], 2)+pow(r1[1][i]+k1
        [1][0]*dt/2-r3[1][i], 2)+pow(r1[2][i]+k1[2][0]*dt/2-r3[2][i], 2)
        ,3/2);
    for (j=0; j<=2; j++) {
        k2[j][0]=k1[j][0]+K1[j][0]*dt/2;
        K2[j][0]=G*m2/d1d2[i]*(r2[j][i]-k1[j][0]*dt/2-r1[j][i])+G*m3/d1d3[
            i]*(r3[j][i]-k1[j][0]*dt/2-r1[j][i]);
    }
    //K3
    d1d2[i]=pow(pow(r1[0][i]+k2[0][0]*dt/2-r2[0][i], 2)+pow(r1[1][i]+k2
        [1][0]*dt/2-r2[1][i], 2)+pow(r1[2][i]+k2[2][0]*dt/2-r2[2][i],2)
        ,3/2);
    d1d3[i]=pow(pow(r1[0][i]+k2[0][0]*dt/2-r3[0][i], 2)+pow(r1[1][i]+k2
        [1][0]*dt/2-r3[1][i], 2)+pow(r1[2][i]+k2[2][0]*dt/2-r3[2][i], 2)
        ,3/2);
    for (j=0; j<=2; j++) {
        k3[j][0]=k2[j][0]+K2[j][0]*dt/2;
        K3[j][0]=G*m2/d1d2[i]*(r2[j][i]-k2[j][0]*dt/2-r1[j][i])+G*m3/d1d3[
            i]*(r3[j][i]-k2[j][0]*dt/2-r1[j][i]);
        }
         //K4
        d1d2[i]=pow(pow(r1[0][i]+k3[0][0]*dt-r2[0][i], 2)+pow(r1[1][i]+k3
            [1][0]*dt-r2[1][i], 2)+pow(r1[2][i]+k3[2][0]*dt-r2[2][i],2),3/2);
        d1d3[i]=pow(pow(r1[0][i]+k3[0][0]*dt-r3[0][i], 2)+pow(r1[1][i]+k3
            [1][0]*dt-r3[1][i], 2)+pow(r1[2][i]+k3[2][0]*dt-r3[2][i], 2),3/2);
        for (j=0; j<=2; j++) {
            k4[j][0]=k3[j][0]+K3[j][0]*dt;
            K4[j][0]=G*m2/d1d2[i]*(r2[j][i]-k3[j][0]*dt-r1[j][i])+G*m3/d1d3[i
                ]*(r3[j][i]-k3[j][0]*dt-r1[j][i]);
            //These is to compute the position and the speed
            v1[j][i+1]=v1[j][i]+0.167*(K1[j][0]+2*K2[j][0]+2*K3[j][0]+K4[j
                ][0])*dt;
            r1[j][i+1]=r1[j][i]+0.167*(k1[j][0]+2*k2[j][0]+2*k3[j][0]+k4[j
                ][0])*dt;
        }
        //K for the second body
        //K1
        d1d2[i]=pow(pow(r1[0][i]-r2[0][i], 2)+pow(r1[1][i]-r2[1][i], 2)+pow(r1
            [2][i]-r2[2][i],2),3/2);
        d2d3[i]=pow(pow(r2[0][i]-r3[0][i], 2)+pow(r2[1][i]-r3[1][i], 2)+pow(r2
            [2][i]-r3[2][i], 2),3/2);
        for (j=0; j<=2; j++) {
            K1[j][1]=G*m3/d2d3[i]*(r3[j][i]-r2[j][i])+G*m1/d1d2[i]*(r1[j][i]-
                r2[j][i]);
            k1[j][1]=v2[j][i];
        }
        //K2
        d1d2[i]=pow(pow(r1[0][i]-r2[0][i]-k1[0][1]*dt/2, 2)+pow(r1[1][i]-r2
            [1][i]-k1[1][1]*dt/2, 2)+pow(r1[2][i]-r2[2][i]-k1[2][1]*dt/2,2)
            ,3/2);
        d2d3[i]=pow(pow(r2[0][i]+k1[0][1]*dt/2-r3[0][i], 2)+pow(r2[1][i]+k1
            [1][1]*dt/2-r3[1][i], 2)+pow(r2[2][i]+k1[2][1]*dt/2-r3[2][i], 2)
            ,3/2);
        for (j=0; j<=2; j++) {
            k2[j][1]=k1[j][1]+K1[j][1]*dt/2;
            K2[j][1]=G*m3/d2d3[i]*(r3[j][i]-r2[j][i]-k1[j][1]*dt/2)+G*m1/d1d2[
                i]*(r1[j][i]-r2[j][i]-k1[j][1]*dt/2);
        }
         //K3
        d1d2[i]=pow(pow(r1[0][i]-r2[0][i]-k2[0][1]*dt/2, 2)+pow(r1[1][i]-r2
            [1][i]-k2[1][1]*dt/2, 2)+pow(r1[2][i]-r2[2][i]-k2[2][1]*dt/2,2)
            ,3/2);
        d2d3[i]=pow(pow(r2[0][i]+k2[0][1]*dt/2-r3[0][i], 2)+pow(r2[1][i]+k2
            [1][1]*dt/2-r3[1][i], 2)+pow(r2[2][i]+k2[2][1]*dt/2-r3[2][i], 2)
            ,3/2);
        for (j=0; j<=2; j++) {
            k3[j][1]=k2[j][1]+K2[j][1]*dt/2;
            K3[j][1]=G*m3/d2d3[i]*(r3[j][i]-r2[j][i]-k2[j][1]*dt/2)+G*m1/d1d2[
                i]*(r1[j][i]-r2[j][i]-k2[j][1]*dt/2);
        }
        d1d2[i]=pow(pow(r1[0][i]-r2[0][i]-k3[0][1]*dt, 2)+pow(r1[1][i]-r2[1][i
            ]-k3[1][1]*dt, 2)+pow(r1[2][i]-r2[2][i]-k3[2][1]*dt,2),3/2);
        d2d3[i]=pow(pow(r2[0][i]+k3[0][1]*dt-r3[0][i], 2)+pow(r2[1][i]+k3
            [1][1]*dt-r3[1][i], 2)+pow(r2[2][i]+k3[2][1]*dt-r3[2][i], 2),3/2);
```

```
for (j=0; j<=2; j++) {
    //K4
    k4[j][1]=k3[j][1]+K3[j][1]*dt;
    K4[j][1]=G*m3/d2d3[i]*(r3[j][i]-r2[j][i]-k3[j][1]*dt)+G*m1/d1d2[i
        ]*(r1[j][i]-r2[j][i]-k3[j][1]*dt);
}
//Compute the second body position and speed
for (j=0; j<=2; j++) {
    v2[j][i+1]=v2[j][i]+0.167*(K1[j][1]+2*K2[j][1]+2*K3[j][1]+K4[j
        ][1])*dt;
    r2[j][i+1]=r2[j][i]+0.167*(k1[j][1]+2*k2[j][1]+2*k3[j][1]+k4[j
        ][1])*dt;
}
//Compute the k for the third body
//K1
d1d3[i]=pow(pow(r1[0][i]-r3[0][i], 2)+pow(r1[1][i]-r3[1][i], 2)+pow(r1
    [2][i]-r3[2][i], 2),3/2);
d2d3[i]=pow(pow(r2[0][i]-r3[0][i], 2)+pow(r2[1][i]-r3[1][i], 2)+pow(r2
    [2][i]-r3[2][i], 2),3/2);
for (j=0; j<=2; j++) {
    K1[j][2]=G*m1/d1d3[i]*(r1[j][i]-r3[j][i])+G*m2/d2d3[i]*(r2[j][i]-
        r3[j][i]);
    k1[j][2]=v3[j][i];
}
//K2
d2d3[i]=pow(pow(r2[0][i]-r3[0][i]-k1[0][2]*dt/2, 2)+pow(r2[1][i]-r3
    [1][i]-k1[1][2]*dt/2, 2)+pow(r2[2][i]-r3[2][i]-k1[2][2]*dt/2, 2)
    ,3/2);
d1d3[i]=pow(pow(r1[0][i]-r3[0][i]-k1[0][2]*dt/2, 2)+pow(r1[1][i]-r3
    [1][i]-k1[1][2]*dt/2, 2)+pow(r1[2][i]-r3[2][i]-k1[2][2]*dt/2, 2)
    ,3/2);
for (j=0; j<=2; j++) {
    k2[j][2]=k1[j][2]+K1[j][2]*dt/2;
    K2[j][2]=G*m1/d1d3[i]*(r1[j][i]-r3[j][i]-k1[j][2]*dt/2)+G*m2/d2d3[
        i]*(r2[j][i]-r3[j][i]-k1[j][2]*dt/2);
}
//K3
d2d3[i]=pow(pow(r2[0][i]-r3[0][i]-k2[0][2]*dt/2, 2)+pow(r2[1][i]-r3
    [1][i]-k2[1][2]*dt/2, 2)+pow(r2[2][i]-r3[2][i]-k2[2][2]*dt/2, 2)
    ,3/2);
d1d3[i]=pow(pow(r1[0][i]-r3[0][i]-k2[0][2]*dt/2, 2)+pow(r1[1][i]-r3
    [1][i]-k2[1][2]*dt/2, 2)+pow(r1[2][i]-r3[2][i]-k2[2][2]*dt/2, 2)
    ,3/2);
for (j=0; j<=2; j++) {
    k3[j][2]=k2[j][2]+K2[j][2]*dt/2;
    K3[j][2]=G*m1/d1d3[i]*(r1[j][i]-r3[j][i]-k2[j][2]*dt/2)+G*m2/d2d3[
        i]*(r2[j][i]-r3[j][i]-k2[j][2]*dt/2);
}
//K4
d2d3[i]=pow(pow(r2[0][i]-r3[0][i]-k3[0][2]*dt, 2)+pow(r2[1][i]-r3[1][i
    ]-k3[1][2]*dt/2, 2)+pow(r2[2][i]-r3[2][i]-k3[2][2]*dt/2, 2),3/2);
d1d3[i]=pow(pow(r1[0][i]-r3[0][i]-k3[0][2]*dt, 2)+pow(r1[1][i]-r3[1][i
    ]-k3[1][2]*dt/2, 2)+pow(r1[2][i]-r3[2][i]-k3[2][2]*dt/2, 2),3/2);
for (j=0; j<=2; j++) {
    k4[j][2]=k3[j][2]+K3[j][2]*dt;
```

```
            K4[j][2]=G*m1/d1d3[i]*(r1[j][i]-r3[j][i]-k3[j][2]*dt)+G*m2/d2d3[i
                ]*(r2[j][i]-r3[j][i]-k3[j][2]*dt);
        }
        for (j=0; j<=2; j++) {
            //Calculate the speed and position for the third body
            v3[j][i+1]=v3[j][i]+0.167*(K1[j][2]+2*K2[j][2]+2*K3[j][2]+K4[j
                ][2])*dt;
            r3[j][i+1]=r3[j][i]+0.167*(k1[j][2]+2*k2[j][2]+2*k3[j][2]+k4[j
                ][2])*dt;
        }
    }
}
```

References

[1] Wolphram alpha for consulting astronomic data

[2] Computational Physics/KONSTANTINOS N. ANAGNOSTOPOULOS